PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-216851
(43)Date of publication of application : 27.08.1993

(51)Int.Cl. G06F 15/16
G06F 9/46

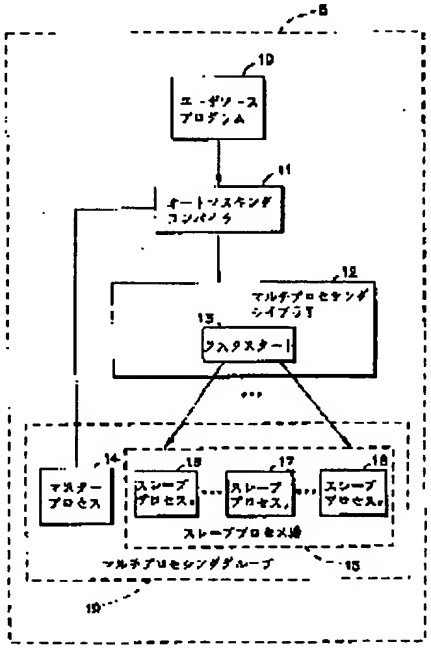
(21)Application number : 03-361081
(22)Date of filing : 19.12.1991

(71)Applicant : CRAY RES INC
(72)Inventor : FURTNEY MARK
BARRIUSO FRANK R
ANDREASEN CLAYTON D
HOEL TIMOTHY W
LACROIX SUZANNE L
REINHARDT STEVEN P

(30)Priority
Priority number : 90 630301 Priority date : 19.12.1990 Priority country : US

(54) METHOD FOR HIGH SPEED COMMUNICATION BETWEEN USER PROGRAM AND OPERATING SYSTEM

(57)Abstract:
PURPOSE: To provide a method for applying the efficient distribution and execution of a parallel task by demanding an additional CPU to operate a task in the parallel area of a code by a master process.
CONSTITUTION: Each thread structure includes a 'wake-up word' written by a master process, and read by an operating system. A master process 14 starts in a single CPU mode at the time of starting execution, and sets the wake-up word of any sleeping slave thread structure a certain non-zero value at the time of coming across the code of a multiple CPU area. Then, a slave process N18 is in a sleeping state, the master process 14 sets the wake-up word a non- zero value, and the operating system asynchronously operates the polling of the wake-up word. A newly scheduled process is executed in a multiprocessing library 12, and a task is scheduled for each process.



LEGAL STATUS

[Date of request for examination] 09.12.1998
[Date of sending the examiner's decision of rejection] 02.04.2002
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平5－216851

(43)公開日　平成５年(1993)８月27日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　15/16	４３０	9190－5Ｌ		
9/46	３６０　Ｂ	8120－5Ｂ		

審査請求　未請求　請求項の数19(全 13 頁)

(21)出願番号　特願平3－361081

(22)出願日　平成３年(1991)12月19日

(31)優先権主張番号　６３０３０１

(32)優先日　1990年12月19日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　592032762
クレイ　リサーチ，インコーポレイティド
アメリカ合衆国，ミネソタ　55121，イーガン，ローン　オーク　ドライブ　655エィ

(72)発明者　マーク　ファートニー
アメリカ合衆国，ミネソタ　55124，アップルバレー，ワンハンドレッドサーティース　ストリート　コート　5830

(74)代理人　弁理士　青木　朗　（外４名）

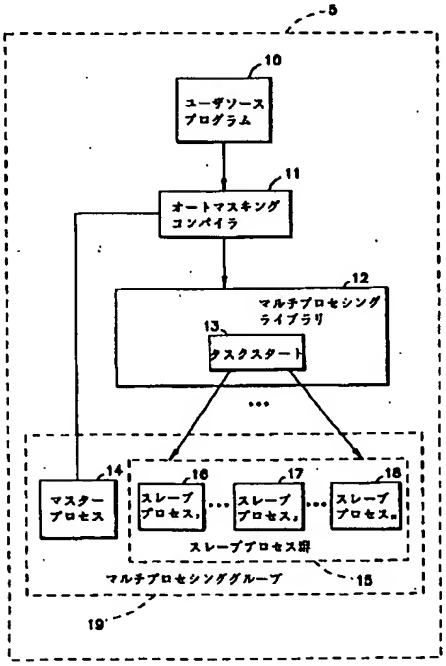
最終頁に続く

(54)【発明の名称】　ユーザプログラムおよびオペレーティングシステム間の高速通信のための方法

(57)【要約】

【目的】　アプリケーションプログラム、マイクロプロセシングライブラリー及びオペレーティングシステム間の通信用効率的プロトコルを提供する。

【構成】　新規プロトコルはウエイクアップ、ギブアップ、及びコンテクスト・トゥ・ユーザ・スペースと称し、主たる用途はマルチＣＰＵを、一つのマルチＣＰＵ、マルチプログラミング、マルチタスキング環境にスケジュールすることであり、ウエイクアップはマスタープロセスがスレーブプロセスの迅速非同期スケジューリングで並列タスクの実行を援助させ、ギブアップはオペレーティングシステムの中断前にスレーブプロセスがタスク終了時にスレーブプロセスの内容をセーブする必要なしに終了する時間を与え、コンテクスト・トゥ・ユーザ・スペースは、他のスレーブプロセスに、ギブアップ下では終了不可の中断されたタスクの実行機会を与える。



1

【特許請求の範囲】

【請求項１】　マイクロプロセッサシステムにおいて、マスタプロセスがスリーピングプロセスを非同期的に起動できるようにし、且つ、コードのマルチＣＰＵ領域に遭遇したとき並列タスクを割り当て可能にする方法であって、

（ａ）該マスタプロセスは追加のプロセスに関係するデータバッファ領域にウエークアップフラグを設定し、

（ｂ）該オペレーティングシステムにより該フラグを非同期的にポーリングし、

（ｃ）該オペレーティングシステムは、ウエークアップフラグを検出すると、スリーピングプロセスを起動し、そして

（ｄ）該起動されたプロセスに対して並列タスクをスケジューリングする、というステップを備える方法。

【請求項２】　マイクロプロセッサシステムにおいて、オペレーティングシステムのタスクが完了するか、又は時間の一部分の満了まで、オペレーティングシステムのプロセスへの割り込みを遅延させる方法であって、

（ａ）該オペレーティングシステムは該プロセスに関係するデータバッファ内にギブアップフラグを設定し、

（ｂ）該ギブアップフラグが設定されると、該プロセスは該ギブアップフラグを読み取り、該ＣＰＵを該オペレーティングシステムに返還し、

（ｃ）該プロセスがその仕事を終了しておらず、且つ、該ステップ（ｂ）にしたがって該オペレーティングシステムに該ＣＰＵを返却していない場合、該オペレーティングシステムは時間の一部分の終わりで該プロセスに割り込む、というステップを備える方法。

【請求項３】　マイクロプロセッサシステムにおいて、並列タスクを割り込みプロセスから利用可能プロセスに割り当てる方法であって、

（ａ）オペレーティングシステムに所与のスレーブプロセスが現在並列タスクを実行中であることを通知するセーブ・イン・ユーザ・スペース・フラグを設定し、

（ｂ）割り込みがあると、該所与のスレーブプロセスの内容をユーザ・スペースにセーブし、

（ｃ）並列タスクを実行中に該所与のタスクが中断されたことをマイクロプロセッサライブラリーに通知するフラグを設定し、

（ｄ）該マイクロプロセッサライブラリーにより該フラグを非同期的にポーリングして中断されたタスクを検出し、そして

（ｅ）該中断されたタスクを第１の利用可能タスクに割り当てる、というステップを備える方法。

【請求項４】　請求項１記載の方法において、更に、起動されたプロセスへのタスクのスケジューリングはマイクロプロセッサライブラリーにより行われる方法。

【請求項５】　請求項１記載の方法において、タスクが完了すると、次のタスクが利用可能な場合次のタスクを

2

起動されたプロセスにスケジューリングする方法。

【請求項６】　請求項５記載の方法において、該次のタスクは、該データバッファに格納されていたコンテクストから切断されたタスクである方法。

【請求項７】　請求項２記載の方法において、更に、プロセスがその割り当てられたタスクを完了した後に、該プロセスは関係するギブアップフラグをチェックする方法。

【請求項８】　請求項３記載の方法において、所与のプロセスをタスクに割り当てるステップは、該プロセスが現在並列タスクを実行していないことにより割り込まれると、オペレーティングシステムに通知して該所与のプロセスの内容をシステム空間にセーブするためのセーブ・イン・システム・スペース・フラグを設定することを含む方法。

【請求項９】　請求項２記載の方法において、さらに各スレーブプロセスはこれと関係しているスレッド構造を有し、オペレーティングシステムが中断を要求するプロセスがそのオペレーティングシステムと関係するスレッド構造を有しているかどうかをオペレーティングシステムがチェックし、もし有していれば、そのプロセスに対してギブアップフラグを設定し、もし有していなければ、そのプロセスを中断する方法。

【請求項１０】　更に、タスクの完了の際に、時間の一部分のスピン待機およびその後のオペレーティングシステムへのそのＣＰＵの解放のプロセスのステップを含む、請求項１記載の方法。

【請求項１１】　請求項２記載の方法において、該方法は更に、

（ｄ）プロセスが並列の仕事を実行中の場合、コンテクストセーブフラグを「セーブ・イン・ユーザ・スペース」に設定し、

（ｅ）コンテクストセーブフラグがそのプロセスを中断する場合はコンテクストセーブフラグをオペレーティングシステムがチェックし、そのフラグが設定されてそのコンテクストが他のプロセスにより検索される場合は中断されたプロセスのコンテクストをユーザースペースにセーブする、というステップを備える方法。

【請求項１２】　請求項１１記載の方法であって、更に、

（ｆ）そのプロセスがセーブに値するコンテクストを有しない場合、コンテクストセーブフラグを「ドント・セーブ・アット・オール」に設定し、

（ｇ）該コンテクストセーブフラグがドント・セーブ・アット・オールに設定されると、オペレーティングシステムが中断されたプロセスのコンテクストをセーブしない、というステップを備える方法。

【請求項１３】　請求項１０記載の方法において、該時間の一部分は、スレーブプロセスが合理的な長さの時間を要求しているタスクを終了するのに充分な長さの期間

のものである方法。

【請求項１４】　請求項２記載の方法において、中断されたプロセスは優先度に基づいて再スケジュールされる方法。

【請求項１５】　請求項３記載の方法であって、更に第１の利用可能プロセスがユーザスペースから中断された所与のタスクのコンテクストを検索し、そのコンテクストを用いてそのタスクの実行を継続するというステップを備える方法。

【請求項１６】　マルチプロセッサシステムにおいて、コードの並列領域の並列処理の方法であって、

（ａ）マスタープロセスと一つ以上のスレーブプロセスを確立し、

（ｂ）該スレーブプロセスの各々に対して、各々がウエイクアップフラグとギブアップフラグを含んでいるスレッド構造とコンテクスト構造とを含むデータバッファコンジットを確立し、

（ｃ）コードの平行領域に遭遇するとマスタープロセスはスレーブプロセスに対してウエイクアップフラグを設定し、

（ｄ）オペレーティングシステムは非同期的に該ウエイクアップフラグをポーリングし、且つ、スレーブプロセスに対してウエイクアップフラグの検出の際に、スレーブプロセスを起動して走らせ、

（ｅ）オペレーティングシステムが走行中のスレーブプロセスを中断したい場合、オペレーティングシステムは走行中のスレーブプロセスに対してギブアップフラグを設定し、

（ｆ）ギブアップフラグが決定されると、走行中のスレーブプロセスはそのタスクの終了の際にギブアップフラグをポーリングし、且つ、制御をオペレーティングシステムに戻し、そして

（ｇ）スレーブプロセスが所定の時間の一部分内に制御をオペレーティングシステムに返還しない場合は、オペレーティングシステムはギブアップフラグを設定したスレーブプロセスを中断するというステップを備える方法。

【請求項１７】　請求項１６記載の方法において、更に、スレッド構造はコンテクストセーブフラグを含み、該方法は更に、

（ｈ）スレーブプロセスが現在並列の仕事を実行中の場合、コンテクストセーブフラグを「セーブ・イン・ユーザ・スペース」に設定し、そして

（ｉ）オペレーティングシステムがスレーブプロセスを中断する場合にオペレーティングシステムはコンテクストセーブフラグをチェックし、且つ、フラグが設定されてコンテクストが他のプロセスにより検索される場合はユーザスペース内の中断されたプロセスのコンテクストをセーブする、というステップを備える方法。

【請求項１８】　請求項１７記載の方法であって、更に、

（ｊ）プロセスが並列の仕事を現在実行中でない場合、コンテクストセーブフラグを「セーブ・イン・システム・スペース」に設定し、

（ｋ）コンテクストセーブフラグがセーブ・イン・システム・スペースに設定されると、オペレーティングシステムは中断されたスレーブプロセスのコンテクストをセーブする、というステップを備える方法。

【請求項１９】　請求項１７記載の方法であって、

（ｌ）プロセスがセーブに値するコンテクストを有しない場合はコンテクストセーブフラグを「ドント・セーブ・アット・オール」に設定し、

（ｍ）コンテクストセーブフラグがドント・セーブ・アット・オールに設定されると、オペレーティングシステムは中断されたスレーブプロセスのコンテクストをセーブしない、というステップを備える方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、一般的にマルチプロセシング・マルチプログラミング・コンピュータシステムにおけるプロセス管理に関する。特に、本発明は、ホストオペレーティングシステムとユーザアプリケーションプログラムとの間の通信に向けられたものである。

【０００２】

【従来の技術および発明が解決しようとする課題】計算システムの出現以来、重要な目標はプログラム実行を高速化することである。単一処理システムにおいては、その好ましい方法は、システムを通る電気信号を高速化するようにより高速な電子構成部品を設計することであった。そのようなシステムにとって、光の速さのような理論的限界は、プログラム実行の速さに対して上限を与える。

【０００３】プログラムコードのいくつかのセクションが実行順序において他のセクションと独立している時、マルチプロセシングは可能である。その場合、多重のＣＰＵ（マルチＣＰＵ）が、これらのコードセクションを同時に実行できる。理想的には、Ｎ台のＣＰＵがプログラムコードを同時に実行するならば、プログラムはＮ倍速く走行するであろう。しかしながら、種々の理論的および実際的な理由のため、この最良の場合は不可能である。

【０００４】理論的に、自明でない部分的順序で実行されねばならないコードセクションが存在するばらば、例えばコードのいくつかのセクションが他のセクションの実行終了を待ち、それらの結果を使用しなければならないならば、プログラムはこの最大の高速化を達成しない。これらのデータ依存は、与えられたプログラムが、そのコードを同時に実行する１台以上のＣＰＵを持つことによって、どのくらい高速化可能であるかを示す。それらはまた、与えられたプログラムが、最大１台のＣＰ

Ｕを必要とするコードセクションと、多重のＣＰＵが使用可能な他のセクションとを通過する可能性があることを示す。与えられたプログラムに関する、単一ＣＰＵモードと多重ＣＰＵモードとの間のこの遷移は、プログラムが一般に最大の高速化を達成しない実際的な理由を作り出す。

【０００５】１つの実際的な理由は、ユーザプログラムとホストオペレーティングシステムとの間の通信を含む。マルチプログラミング環境（すなわち、いくつかのユーザプログラムが計算リソースに関して同時に競合する）においては、１つのプログラムが単一ＣＰＵモードにある時に、多重ＣＰＵがそれに結びつけられたままにしておくことは通常、非効率的である。これらアイドルのＣＰＵは、そのプログラムが多重ＣＰＵモードにリターンするまで、他のプログラム上で働くことによってシステムスループットにさらに良く奉仕できるであろう。しかしながらプログラムが多重ＣＰＵモードにリターンする時に、プログラムはオペレーティングシステムから追加のＣＰＵを要求する必要がある。従来技術においては、ユーザプログラムおよびオペレーティングシステムは、歴史的に通信のための２つの道を有している。すなわち、ユーザープログラムによるサービスの要求であるシステムコールと、オペレーティングシステムがユーザプログラムに特定の情報を報告する機構である割込みとである。どちらの機構も、マルチプログラミング環境における効率的なマルチタスキングに必要な高速通信を提供しない。高速通信の方法がないと、計算システムは、最大の高速化または効率的なシステムスループットのどちらかを達成するにはほど遠い。

【０００６】ユーザプログラムが追加のＣＰＵに対して充分に小さなコードセクションを処理するよう要求する時、問題が発生する。悪い場合には、単一ＣＰＵがコードセクション全体を実行するのに要する時間は、余分なＣＰＵに援助を要求するのに要する時間と等しいかまたはそれよりも小さくなるであろう。システムスループットは、これらの要求を処理することによって減少する。また、プログラムの実行は、プログラムが追加のＣＰＵに対し要求しかつ待つためにおそくなるならば、害を受ける。結果として、利用可能な並列化は比較的雑なプログラム構造上で発生しなければならず、マルチプロセシングのための機会は失われる。

【０００７】オペレーティングシステム割込みについて、他の問題が発生する。もしもオペレーティングシステムが、現在優先順位がより高いプロセスを接続させるために、有効な並列作業を実行しているプロセスを分離する必要があるならば、分離されるプロセスのコンテクスト（内容）はセーブされねばならない。この割込みは、２つの非効率を導く。

【０００８】第１に、コンテクストのセーブは追加のオーバヘッドである。しかしながら、コンテクストのセーブおよび回復は、いくつかの型式の並列作業にとって必要でない。例えば、並列ＤＯループの繰り返しを実行する仕事を課せられたプロセスが、オペレーティングシステムによって割込まれる場合を考える。この割込みが特定の繰り返しの終わりに発生するならば、そのプロセスはその有効な作業を終了させそしてその結果を共通メモリに戻してしまっているであろう。その場合には、後にそのプロセスをリスタートするために、何もコンテクストはセーブされる必要がない。

【０００９】第２に、ユーザプログラムは、割込まれたプロセスが有効な作業を続ける前に戻るのを待たねばならないかもしなれない。一般に、並列領域における作業全てが完了するまで、並列領域を越える作業は開始されえない。これは、プログラムの正確さを保証するために必要である。割込まれたプロセスがその作業を続けるために戻っていなければ、並列作業を終えた他のプロセスは待つように強制される。

【００１０】

【課題を解決するための手段および作用】本発明は、マスタープログラム（例えば、ユーザプログラムを実行するプロセス）、オペレーティングシステム、およびスレーブプロセスをスケジュールして「タスク」と称する並列の仕事の一部分を実行するマルチプロセシングライブラリーの間で、高速、非同期通信のための３つのフロトコルを提供することにより上記の問題を解決する。これらのプロトコルは「コンジット」と称するアグリードアポンデータバッファを使用する。

【００１１】第１のプロトコルは「ウエイクアップ」であり、マスタープロセスが追加のＣＰＵにコードの並列領域で仕事をするように要求することを可能にする。マスタープロセスは、並列領域を検出すると、スレーブプロセスと関係しているコンジット内のフラグを設定する。オペレーティングシステムは非同期的にこれらのフラグをポーリングし休眠中のスレーブプロセスを目覚めさせる。目覚めたスレーブプロセスはマルチプロセシングライブラリーにより順次スケジュールされて並列タスクを実行する。

【００１２】第２のプロトコルは「ギブアップ」であり、オペレーティングシステムがタスクを切断する前にそのタスクを完了するのに充分な時間をスレーブプロセスに与える。盲目的に切断する代わりに、オペレーティングシステムはマイクロプロセシングランブラリーが要求として順次読み出すコンジット内にフラグを設定してスレーブプロセスに所属している特定のＣＰＵを戻す。そのＣＰＵに所属しているプロセスがその仕事（例えば、並列ＤＯループの繰り返し）を終了すると、他のタスク割当のためにプロセスはマイクロプロセシングライブラリーに戻る。そのプロセスに他のタスクを与える代わりに、マイクロプロセシングライブラリーはＣＰＵをオペレーティングシステムに戻す。スレーブプロセスは

そのコンテクストをセーブすることなしに休眠状態になる。

【0013】第3のプロトコルは、コンテクスト・トゥー・ユーザー・スペースであり、オペレーティングシステムが中断の前にタスクを終了するためにスレーブプロセスを待つことができないときに採用される。この場合、中断されたプロセスのコンテクストはシステムスペースではなくてユーザースペースにセーブされる。オペレーティングシステムはコンジットの中にフラグを設定する。マイクロプロセシングライブラリーは後に、有用な仕事中に中断されたプロセスとしてそのフラグを解釈する。マイクロプロセシングライブラリーは、中断されたプロセスの仕事を追加のタスク割当のために戻る第1の利用可能スレーブプロセスに割り当てる。

【0014】

【実施例】図面において、類似した数字はいくつかの図を通して類似した要素を指している。

【0015】好ましい実施例の次の詳細な説明においては、本願の一部を形成する添付図面が参照され、図面には本発明が実施される特定の実施例が図示によって示される。他の実施例も利用可能であり、本発明の範囲から逸脱することなしに実現変更がなされてもよいことは理解されるべきである。

【0016】本発明は、3つの別個のプロトコルすなわちウェークアップ、ギブアップおよびコンテクスト・トゥ・ユーザ・スペースを表わす。これらのプロトコルは、協同並列インタフェースの切離せない部分を形成する。そのインタフェースを通して、プロトコルは、ユーザプログラムおよびホストオペレーティングシステム間の高速通信を可能にする。

【0017】好ましい実施例は、ユーザプログラムのマスタープロセスに対してスレーブとなるプロセスのスケジューリングを制御する分離したマルチプロセシングライブラリを利用する。加えて、分離したデータバッファであるコンジット（ｃｏｎｄｕｉｔ）は、ユーザプログラム、マルチプロセシングライブラリ、およびホストオペレーティングシステム間の非同期通信を容易にする。

【0018】図1は、どのようにマルチプロセシンググループが形成されるかを図示する。点線5は、本発明のソフトウェアおよびデータ構成要素が機能するマルチプロセシングシステムを概略的に表示する。明瞭には示されていないが、システム5がプロセッサ間通信のために使用される共有メモリを含むことは理解されるであろう。本発明は特定型式のマルチプロセシングシステムに限定されないが、それが機能する既知のシステムはクレイX－MPおよびY－MPマルチプロセッサシステムを含む。これらのシステムの説明は、例えば、“Ｃｏｍｐｕｔｅｒ　Ｖｅｃｔｏｒ　Ｍｕｌｔｉｐｒｏｃｅｓｓｉｎｇ　Ｃｏｎｔｒｏｌ”という題で1987年1月13日発行の米国特許第4636942号、“Ａｐｐａｒａｔｕｓ　ａｎｄ　Ｍｅｔｈｏｄ　ｆｏｒ　Ｍｕｌｔｉｐｒｏｃｅｓｓｏｒ　Ｃｏｍｍｕｎｉｃａｔｉｏｎ”という題で1988年6月28日発行の米国特許第4754398号、および“Ｓｙｓｔｅｍ　ｆｏｒ　Ｍｕｌｔｉｐｒｏｃｅｓｓｏｒ　Ｃｏｍｍｕｎｉｃａｔｉｏｎ”という題で1990年2月9日提出の米国特許出願第308401号にみられる。ここで使用される「システム・スペース」という語は、オペレーティングシステムのために確保されたメモリ部分を示し、また「ユーザ・スペース」という語は、ユーザプログラムによる使用に割り付けられるスペースを示す。ユーザソースプログラム10は、ユーザソースプログラム10内で発生する並列処理の領域を捜すコンパイラ11を自動的に並列化することによって、好ましくはコンパイルされる。そのコンパイラはまた、自動的に並列化を検出するのと同様にユーザがマニュアルで並列領域を指定する機構を含む。しかしながら、ユーザプログラムがタスクの並列処理としてコンパイルされる方法は、本発明の一部でもなければ本質でもない。

【0019】マルチプロセシングライブラリ12は、ユーザプログラムがジョブ開始時に必要とする追加のプロセス全てを作成する。ライブラリルーチンであるタスクスタート13は、全ての必要なスレーブプロセス15を生成するためにシステムコールを使用する。好ましい実施例は、各物理的ＣＰＵに関して1つのスレーブプロセスを生成する。しかしながら、任意の数のスレーブプロセスで十分であることは理解されるべきである。これらのプロセスは、マスタープロセスとともに同一のマルチプロセシンググループ19に属する。

【0020】図2～図4は、本発明を実現するために使用されるデータ構造を示す。図2において、マルチプロセシングライブラリ12は、ユーザ・スペースにおいて通信を容易にするため、コンジットと呼ばれる、合意に基づくデータバッファを実現する。コンジット20は、個々のスレッド（ｔｈｒｅａｄ）構造およびコンテクスト構造を含む。例えば、スレーブプロセスNは、符号21を付されたスレッド構造と、符号22を付されたタスクコンテクスト構造とを持つ。マルチプロセシングライブラリ12は、ジョブプログラム開始時、マルチプロセシンググループ19における各個々のプロセスに関するスレッド構造を結合させる。マルチプロセシングライブラリ12はシステムコールを用いてこの結合を作り、その結合はユーザプログラムの実行中、持続する。

【0021】図3は、個々のプロセススレッド構造21の構成を示す。各スレッド構造は次の型式のデータを含む。

・ライブラリの要求およびポインタ
・オペレーティングシステムの要求およびポインタ
・ライブラリ・スタティスティクス（統計）
・オペレーティングシステム・スタティスティクス

・トレース情報
各スレッド構造内には、特定のフラグワードすなわちウェークアップワード、ギブアップワードおよびコンテクストセーブフラグが存在する。ウェークアップワードは、マスタープロセス１４によって設定される。ギブアップワードは、オペレーティングシステムによって設定される。

【００２２】プログラム開始時、マスタープロセス１４は、単一ＣＰＵモードで実行を始め、並列コードセクションが出現するまで続く。マスタープロセス１４が単一ＣＰＵモードにある間、スレーブプロセス１５は、「スリープ」状態にある。マスタープロセス１４が多重ＣＰＵモードセクションに出くわす時、マルチプロセシングライブラリ１２に記録されたタスク（すなわち並列作業）が実行のためにスレーブプロセスに与えられる。スレーブプロセスおよびタスクは、ジョブ開始時に作成される。マスタープロセスが並列領域に出くわすまで、プロセスは「スリープ」状態にあり、タスクは「アイドル」である。作成される各タスクは、コンジット２０に置かれる結合コンテクスト構造２２（図４）を持つ。

【００２３】図４は、タスクコンテクスト構造の構成を示す。各タスクは、次の型式のデータを含む。
・私用データ格納領域４１
・ユーザレジスタと関連特権レジスタとステータスワードの全集合を保持するためのスペース
・オペレーティングシステム通信および状態フラグ
・他の雑多なタスク格納領域
タスク私用データ領域４１を別にして、タスクコンテクスト構造２２は、マルチプロセシングライブラリ１２およびオペレーティングシステムによる使用のために確保される。

【００２４】コンジット２０は、プロセスからタスクへのスケジューリング変化のための機構を提供する。このスケジューリング変化は、協同並列インタフェースの３つの主要プロトコル、すなわちウェークアップ、ギブアップおよびコンテクスト・トゥ・ユーザ・スペースによって定義される。好ましい実施例は上述のデータ構造を使用しているが、それらのプロトコルを実現するために他のデータ構造が同様に使用可能であることが理解される。

【００２５】図５は、ウェークアッププロトコルを図示する。ウェークアップは、マスタープロセスが多重ＣＰＵ領域に出くわし追加のＣＰＵを要求する時に発生する問題を回避する。ウェークアップによりマスタープロセスは、早急にコンジット内にフラグを設定し、プログラムコードを直列に実行し続けることができる。オペレーティングシステムは、非同期にフラグを読み取り、要求を処理する。

【００２６】各スレッド構造は、マスタープロセスによって書き込まれオペレーティングシステムによって読み取られる「ウェークアップワード」を含む。上述のように、マスタープロセス１４は、実行を開始する時、単一ＣＰＵモードでスタートする。実行中でないスレーブプロセスは、「スリープ」状態にある。マスタータスクは、ユーザプログラムの直列部分を実行するタスクである。マスタープロセス１４は、多重ＣＰＵ領域のコードに出くわす時、どのスリーピングスレーブスレッド構造のウェークアップワードをもある非零の値に設定する。図５に示されるように、スレーブプロセスＮ１８はスリーピングしており、マスタープロセス１４はそのウェークアップワード５０を非零の値に設定する。オペレーティングシステム５１は、非同期にウェークアップワードをポーリングする。スレーブＮのウェークアップワード５０は非零なので、スレーブプロセスＮ１８は「起こされ」て走行キューに置かれる。

【００２７】新しくスケジュールされたプロセスは、マルチプロセシングライブラリ１２において実行を始める。マルチプロセシングライブラリ１２は、次いで、各起こされたプロセスに対しタスクをスケジュールする。マスタープロセスは、スレッド構造にウェークアップワードを設定する時、またどのコンテクスト（またはタスク）が並列作業に加わるためにスケジュールされるべきかを示すものを、ライブラリによって読み取られるワードに書き込む。タスクはかくして並列作業に加わるべくスケジュールされる。タスクは、並列作業の部分を完了した時、マスチプロセシングライブラリ１２に戻り、「アイドル」状態に置かれる。

【００２８】もしもマルチプロセシングライブラリ１２がプロセスをスケジュールするための他のタスクを持っていなければ、プロセス回転は「スレーブホールド時間」と呼ばれる設定時間の間、ユーザ・スペースにおいて待機している。マルチプロセシングライブラリ１２において待機しているプロセス回転はコンジット２０に格納されたコンテクストと切り離されているタスクが存在するか、またはまだスケジュールされるべきタスクが存在するか、をみるために検査する。もしもそうであれば、プロセスはタスクの実行を開始する。もしもスケジュールされるべきそのようなタスクが存在しないならば、プロセスは、ウェークアップワード５０をクリアーし、スレーブホールド時間の終了時にシステムコールによってそのＣＰＵをギブアップする。

【００２９】図６は、ギブアッププロトコルを示す。ギブアップは、タスクの中ごろにあるプロセスを切り離すのに関連する問題を避ける。ギブアップは、コンテクストのセーブ数を最小化するために、タスクが完了するまで切り離しを延期する。

【００３０】各スレッド構造は、オペレーティングシステム５１によって書き込まれマルチプロセシングライブラリ１２によって読み取られる、「ギブアップワード」６０を含む。加えて、各スレッド構造は、マルチプロセ

シングライブラリ１２によって書き込まれオペレーティングシステム５１によって読み取られる「コンテクストセーブフラグ」６１を含む。このコンテクストセーブフラグは、３つの設定値すなわち「セーブ・イン・ユーザ・スペース」、「セーブ・イン・システム・スペース」およびホールドループにある場合の「ドント・セーブ・アット・オール」を持つ。本質的に、コンテクストセーブフラグは、関連プロセスがタスクを割り当てられる場合そしてその場合のみ、セーブ・イン・ユーザ・スペースに設定される。最初の２つの設定値は、オペレーティングシステム５１に、割込まれる時にこのプロセスのためのレジスタの集合をどこにセーブすべきかを教え、最後の設定値は、何のセーブも実行されるべきでないことを教える。

【００３１】オペレーティングシステム５１は、プロセスに割込む前に、プロセスがそれに関連するスレッド構造を持つかどうかをみるために検査する。もしも持っていなければ、プロセスは切り離される。もしも持っていれば、オペレーティングシステム５１は、コンテクストセーブフラグが「セーブ・イン・ユーザ・スペース」に設定されているかをみるために検査する。もしも設定されていなければ、プロセスは現在何も並列作業をしておらず、切り離されてもよい。もしもセーブ・イン・ユーザ・スペースに設定されていれば、プロセスは現在並列作業をしている。この場合、オペレーティングシステム５１は、ギブアップワード６０を、プロセスが都合のつき次第ＣＰＵをギブアップすべきであることをプロセスに示す負値に設定する。

【００３２】しかしながら、ギブアップ要求についてのオペレーティングシステム５１の忍耐力は有限である。もしもプロセスが与えられた時間内にＣＰＵをギブアップしないならば、オペレーティングシステム５１はそのプロセスに割込む。この時間は任意であるが、プロセスが適度なタスクを終えることができるよう十分大きくなければならない。

【００３３】プロセスがそのタスクを完了しリターンする時、マルチプロセシングライブラリ１２はギブアップワード６０を検査する。もしもそれが負値に設定されていれば、プロセスはすぐにオペレーティングシステム５１に戻って再スケジュールする。ギブアップ要求に関して再スケジュールする時、プロセスはそのウェークアップワード５０をクリアーしない。これは、プロセスがスレーブホールド時間の間待ちを回転させるために、オペレーティングシステムがプロセスを再結合するように強制する。これは、プロセスがオペレーティングシステムに対し再スケジュールする時と異なっており、なぜならばスレーブホールド時間が切れてしまっているからである。

【００３４】図７は、コンテクスト・トゥ・ユーザ・スペースプロトコルを示す。コンテクスト・トゥ・ユーザ・スペースは、切り離しが避けられず有効な並列作業中に発生せざるをえない時に起こる問題を回避する。先に述べたように、並列領域の全作業が完了するまで、並列領域を越える作業は開始できない。これは、プログラムの正確さを保証する。並列作業の中ごろにあるプロセスを切り離すことによって、他の全プロセスは、切り離されたプロセスが再スケジュールされその部分の仕事を完了するまで待たなければならなくなる。コンテクスト・トゥ・ユーザ・スペースは、マルチプロセシングライブラリ１２が切り離されたタスクを、接続されていて仕事を探している他のプロセスにスケジュールするのを可能にするプロトコルである。

【００３５】プロセスが、マルチプロセシングライブラリ１２において実行中でありタスクをスケジュールされたプロセスにある間、コンテクストセーブフラグ６１は「セーブ・イン・システム・スペース」に設定される。この状態においてプロセスが割込まれる場合、そのプロセスのためのレジスタはシステムスペースにセーブされる。ユーザプログラムは、このメモリ部分にアクセスできない。

【００３６】マルチプロセシングライブラリ１２は、プロセスがタスクを実行できるようにする前、プロセスにタスクをスケジュールするやいなや、コンテクストセーブフラグ６１を「セーブ・イン・ユーザ・スペース」に設定する。この設定は、オペレーティングシステム５１に対して、そのプロセスが現在並列領域において実行中であることを示す。オペレーティングシステム５１は、セーブ・イン・ユーザ・スペースに設定されたコンテクストセーブフラグを有するプロセスに割込む時、現在そのプロセスに接続されている、そのタスクのコンテクスト構造７１内に、ユーザレジスタをセーブする。次いでオペレーティングシステム５１は、コンテクスト構造内のインディケータを設定し、マルチプロセシングライブラリ１２に対してこのタスクが並列領域において割込まれたことを表わす。これによって、マルチプロセシングライブラリ１２は、割込まれたタスクを次の利用可能なプロセスにスケジュールすることができる。

【００３７】タスクの完了とライブラリへのリターンに基づいてただちに、セーブフラグはセーブ・イン・システム・スペースに設定される。プロセスは、ひとたびスケジュール可能なタスクが存在しないことを決定すると、セーブフラグをドント・セーブ・アット・オールに設定し、スレーブホールドループにおいて回転する。スレーブフラグがドント・セーブ・アット・オールに設定される唯一の時は、プロセスが、ホールドループにおいて回転しているか、あるいは割込まれたプロセスのコンテクストをセーブし回復するよりも他のプロセスが進行中の作業を再び開始する方がより速いようなライブラリ部分で他の作業をしている時である。特に、タスクがスケジュール可能になるのを待つホールドループにプロセ

スがあるならば、割込まれた場合にそのレジスタをセーブする理由はない。そのレジスタは何も重要なものを保持していないので、レジスタをセーブし回復する作業および遅延を通過する必要はない。ドント・セーブ・アット・オールフラグは、何のコンテクストもセーブされる必要がないことを示す。代わりに、オペレーティングシステムがプロセスを切り離す時、プログラムカウンタレジスタはマルチプロセシングライブラリにおいて実行を始めるように設定され、交換パッケージの残りは零にされ、そしてプロセスは走行キューの底に置かれる。プロセスは、最終的にオペレーティングシステムによって再びスケジュールされる時、マルチプロセシングライブラリにおいて実行を始める。

【００３８】

【発明の効果】かくして、本発明は、マルチプロセシングシステムの効率的使用を容易にし、並列プロセスにおける並列タスクの効率的分布および実行を与える方法を提供する。

【図面の簡単な説明】

【図１】マルチプロセシンググループの作成を示すブロック図である。

【図２】コンジットの構成部品を示す。

【図３】コンジットにおけるプロセススレッド構造の構成要素を示す。

＊【図４】コンジットにおけるタスクコンテクスト構造の構成要素を示す。

【図５】ウェークアッププロトコルを示す。

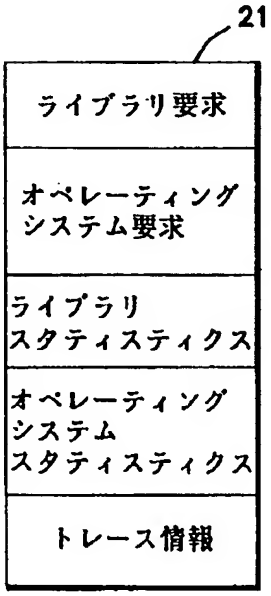
【図６】ギブアッププロトコルを示す。

【図７】コンテクスト・トゥ・ユーザ・スペースプロトコルを示す。

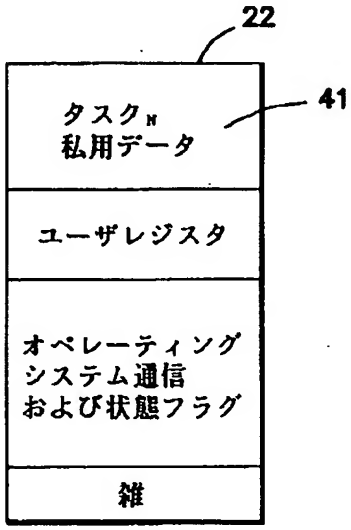
【符号の説明】

５…マルチプロセシングシステム
１０…ユーザソースプログラム
１１…オートマスキングコンパイラ
１２…マルチプロセシングライブラリ
１３…タスクスタートルーチン
１４…マスタープロセス
１５…スレーブプロセス群
１６，１７，１８…スレーブプロセス
１９…マルチプロセシンググループ
２０…コンジット
２１…スレーブプロセススレッド構造
２２…タスクコンテクスト構造
４１…タスク私用データ
５０…スレーブウェークアップワード
５１…オペレーティングシステム
６０…スレーブギブアップワード
６１…スレーブセーブフラグ

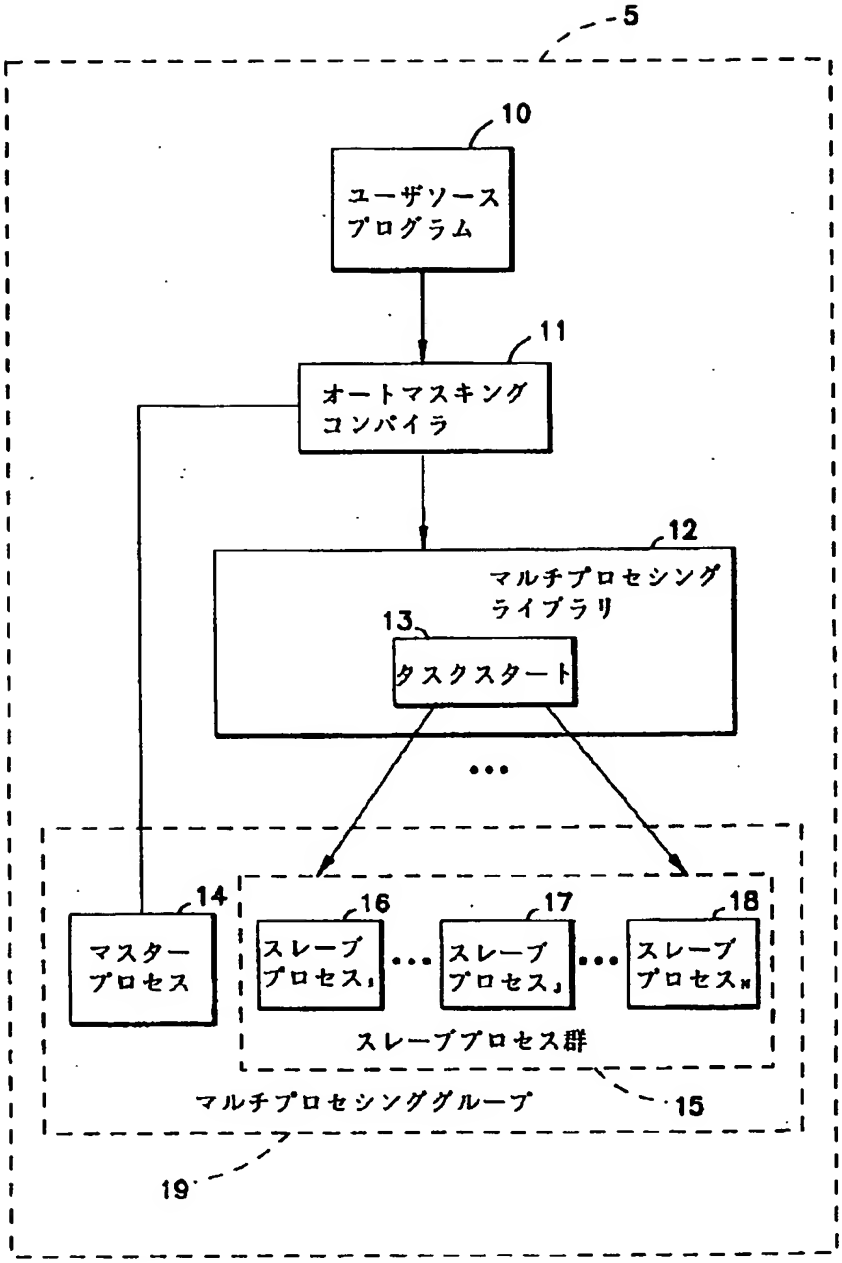
【図３】



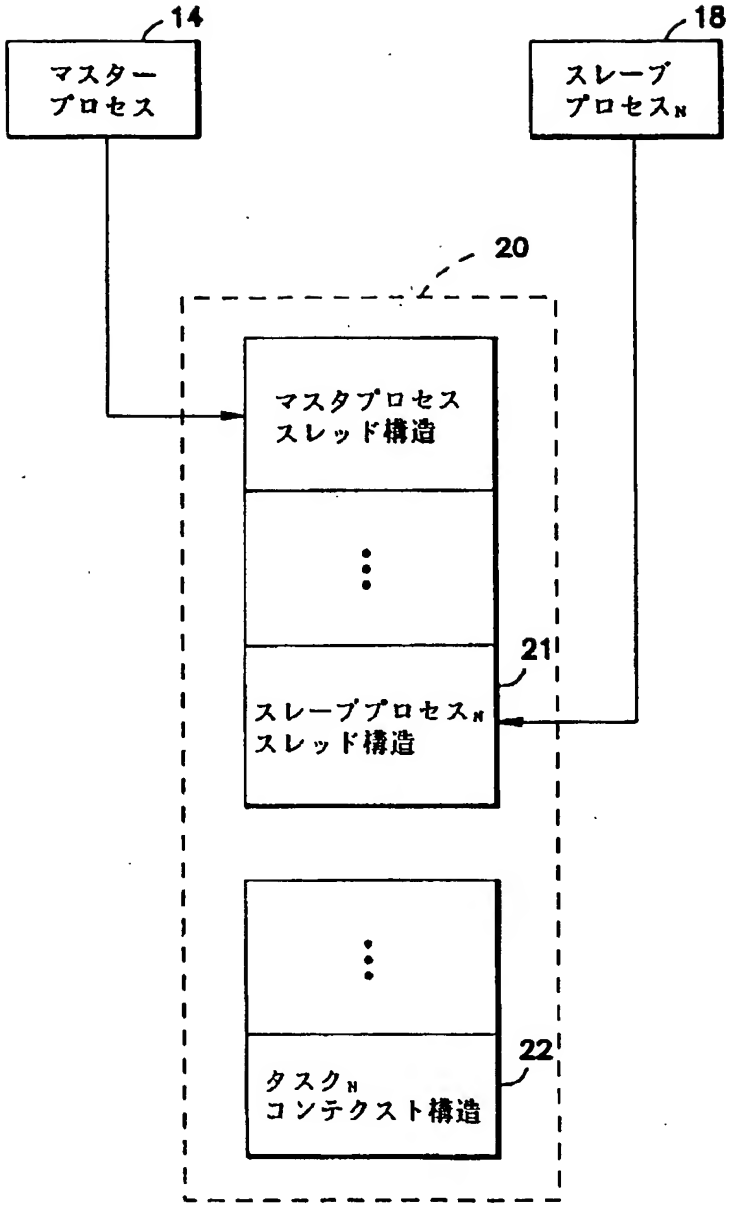
【図４】



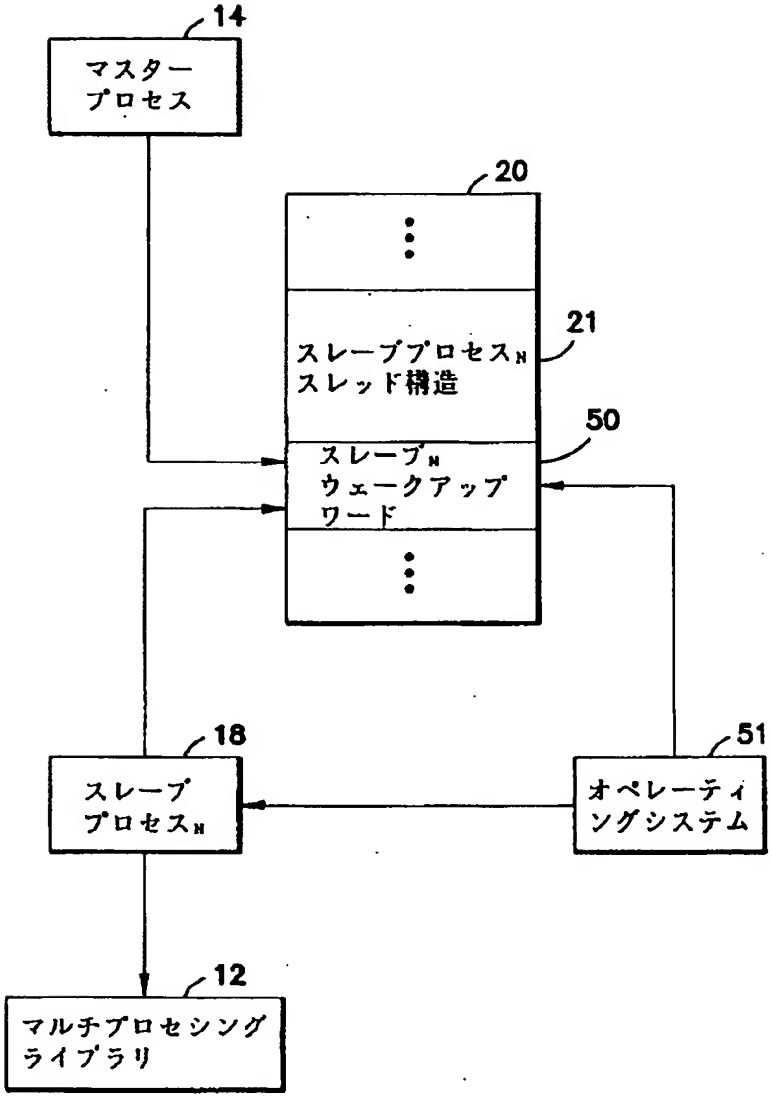
【図1】


5
10
ユーザソース
プログラム
11
オートマスキング
コンパイラ
12
マルチプロセシング
ライブラリ
13
タスクスタート
14
マスター
プロセス
16
スレーブ
プロセス1
17
スレーブ
プロセス2
18
スレーブ
プロセスN
スレーブプロセス群
マルチプロセシンググループ
15
19

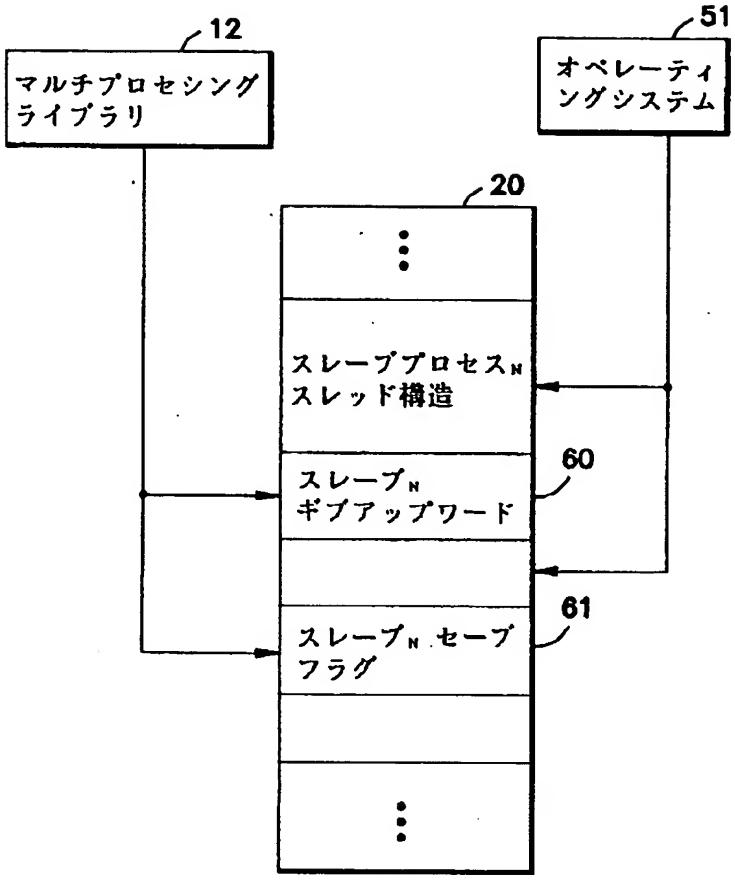
【図2】


14
マスター
プロセス
18
スレーブ
プロセスN
20
マスタプロセス
スレッド構造
21
スレーブプロセスN
スレッド構造
22
タスクN
コンテクスト構造

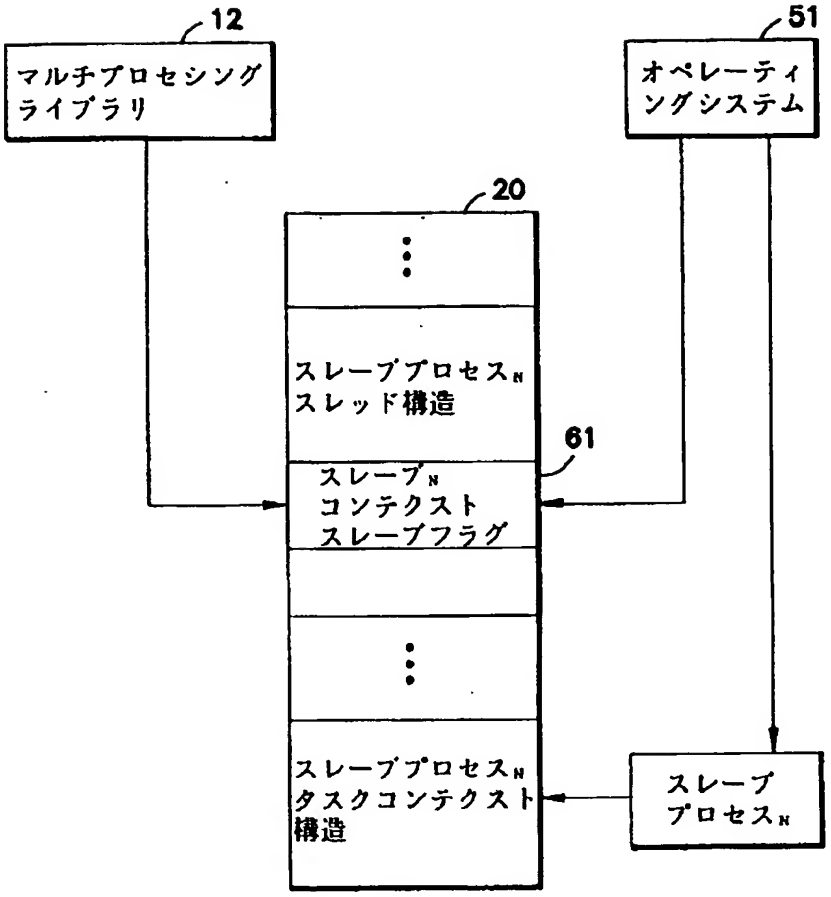
【図5】


14
マスター
プロセス
20
21
スレーブプロセスN
スレッド構造
50
スレーブN
ウェークアップ
ワード
18
スレーブ
プロセスN
51
オペレーティ
ングシステム
12
マルチプロセシング
ライブラリ

【図6】


12
マルチプロセシング
ライブラリ
51
オペレーティ
ングシステム
20
スレーブプロセスN
スレッド構造
60
スレーブN
ギブアップワード
61
スレーブN セーブ
フラグ

【図7】



フロントページの続き

(72)発明者　フランク　アール．バリューソ
アメリカ合衆国，ミネソタ　55124，アップルバレー，ファインドレイ　アベニュ　13285
(72)発明者　クレイトン　ディー．　アンドレアセン
アメリカ合衆国，ミネソタ　55068，ローゼマウント，シャノン　パークウェイ　12800
(72)発明者　ティモシー　ダブリュ．ホーエル
アメリカ合衆国，ミネソタ　55123，イーガン，パーク　リッジ　ドライブ　4667
(72)発明者　スザンヌ　エル．ラクロワ
アメリカ合衆国，ミネソタ　55331，ショアーウッド，マッキンレイ　プレイス　5920．
(72)発明者　スティーブン　ピー．ラインハルト
アメリカ合衆国，ミネソタ　55124，イーガンブリドル　リッジ　ロード　716